

Alternative Least Mean Square Adaptive Filter Architectures for Implementation on Field Programmable Gate Arrays

Sinead Mullins, Conor Heneghan

Digital Signal Processing Group,

Department of Electronic and Electrical Engineering,

University College Dublin,

Dublin 4, Ireland

Tel: +353 1 7161909; fax: +353 1 2830921

e-mail: sinead.mullins@ee.ucd.ie, conor.heneghan@ucd.ie

ABSTRACT

The Least Mean Square (LMS) adaptive filter is a simple well behaved algorithm which is commonly used in applications where a system has to adapt to its environment. Architectures including the direct, transposed and hybrid forms are examined in terms of the following criteria: speed, power consumption and FPGA resource usage. Both the transposed and hybrid forms, which are derived from the delayed LMS, allow for higher speeds without significant increases in power or area. Results for both these adaptations are independent of filter length with the maximum speed of the 16 tap transposed form being over 4 times greater than the speed of a 16 tap direct form implementation. For FPGA implementation, the transposed form is optimal, as power and area are not significantly greater than values found for the direct form, despite the higher maximum frequency. Even at greater numbers of taps, the maximum frequency of the transposed form is not degraded, despite the input data bus driving an increased number of multipliers.

1 Introduction

A core feature of many modern communication systems is their ability to adapt to their working environment. The technology at the heart of these flexible systems is an adaptive digital filter i.e. a digital filter in which the coefficients change in response to external cues. To date, such arithmetic datapaths have been implemented using either digital signal processors (DSPs) or application specific integrated circuits (ASICs). However as technology has advanced, it is now conceivable to implement high performance arithmetic datapaths on devices such as Field Programmable Gate Arrays (FPGA). This paper examines some possible architectures of adaptive filters using the LMS algorithm on the Xilinx Virtex family of FPGA's.

Figure 1 shows a block diagram of how an adaptive filter can be formulated in an equalizer setting. In this case,

the filter w is adapting to produce an output sequence $\hat{d}[n]$ which is identical to a known output $d[n]$. The filter w as being of FIR type, with p coefficients, i.e.

$$w_n = [w[1]w[2]...w[p]]^T \quad (1)$$

The subscript n indicates that the filter coefficients themselves vary with time. The adaptation algorithm calculates the update based on knowledge of the input, and on an error signal $e[n]$. During training, such a signal can be generated by using a known training sequence at both receiver and transmitter.

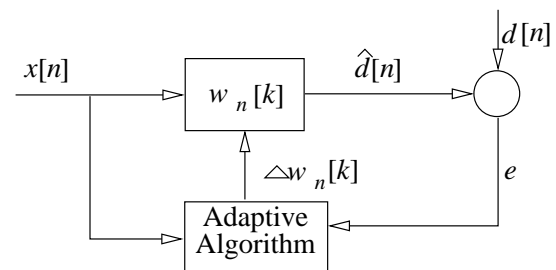


Figure 1: Block diagram of an adaptive filter

The Least Mean Square (LMS) Algorithm is the most widely used technique to find an update equation for the system shown in Figure 1.

For the LMS algorithm, the coefficient vector update equation becomes:

$$w_{n+1} = w_n + \mu e[n]x[n] \quad (2)$$

where $e[n] = \hat{d}[n] - d[n]$, and μ is a scalar called step-size.

The LMS algorithm is a well-behaved algorithm which provides an optimal solution under relatively loose mathematical conditions. A further practical advantage (which explains its widespread use in communication

systems) comes from the fact that the update for the k th coefficient requires only one multiplication and one addition.

2 Architectures for FIR filters

Finite Impulse Response (FIR) filters have two canonical forms called the direct and transposed form [7, 6], illustrated in Figures 2 and 3, where a , b , c , and d are fixed coefficients.

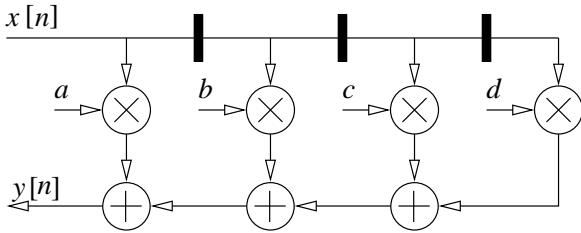


Figure 2: Direct form implementation

In this form the limiting factor is whether all multiplications and additions can be achieved in a single clock cycle (allowing for setup and settling delays). The critical path scales linearly as a function of the number of taps, so for long filters the maximum clock frequency is severely limited.

The critical path of this filter can be reduced by a transformation technique known as retiming, (i.e. moving the delay elements in a circuit without changing the input-output characteristics [7]). Applying this to Figure 2 gives the following transposed implementation shown in Figure 3.

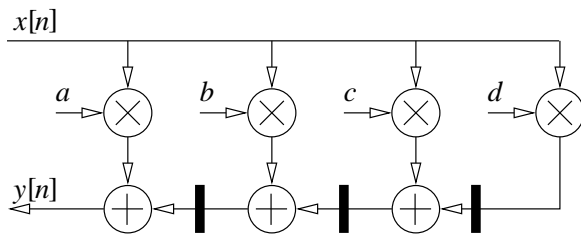


Figure 3: Transposed form implementation

In this architecture, the critical path is reduced to a single multiply accumulate. A potential problem is that the data bus has to drive multiple inputs and the capacitance of the data bus can limit performance.

It is possible to achieve modularity and avoid the critical path limitations of the canonical forms by using so called hybrid architectures [7, 5, 1], where the delay registers are distributed between the data output and input

branches. A basic three tap module of a hybrid form is shown in figure 4. The modularity in hybrid filters arises from the fact that they can be built using a pipeline of identical stages. A hybrid filter of this type has additional benefits in that it has zero latency.

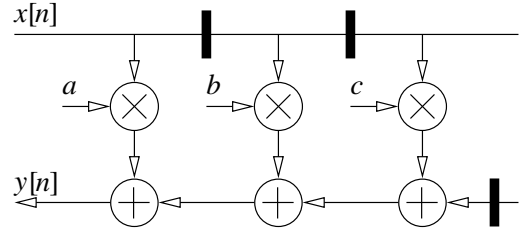


Figure 4: Hybrid form 1

3 Architectures for filters using LMS

The standard LMS algorithm as outlined in equation 2 can be implemented as shown in Figure 5.

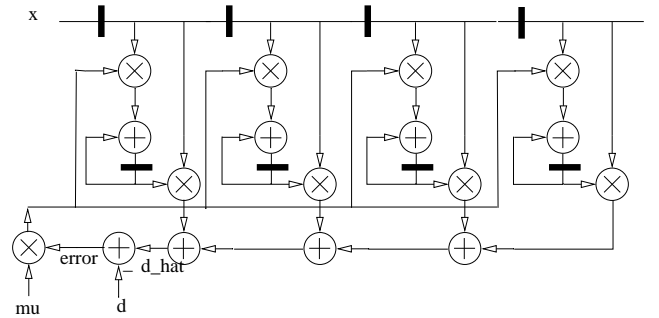


Figure 5: Standard LMS Implementation

The above canonical realizations of an FIR filter, although functionally equivalent, do not perform the same operations when the coefficients vary, so simple retiming as in the fixed coefficient case does not work. Pipelining of this filter is made difficult due to the coefficient feedback loop [7]. An implementation of the LMS algorithm which can be retimed and pipelined to give functionally equivalent representations is the Delayed LMS (DLMS) algorithm. This is implemented by inserting registers along the filter and error feedback path [3, 2]. The DLMS removes some of the restrictions of the conventional LMS algorithm. It slightly increases the training time, but the benefit is that it allows the retiming principle to be applied. The delayed LMS weight update equation is:

$$w(n+1) = w(n) + \mu e(n-D)x(n-D) \quad (3)$$

An architectural implementation of this is shown in Figure 6 where the algorithm is split into the weight update

block and filter block. Applying retiming to this as was done in the case of the FIR filters gives the implementation shown in Figure 7. This transposed form has a

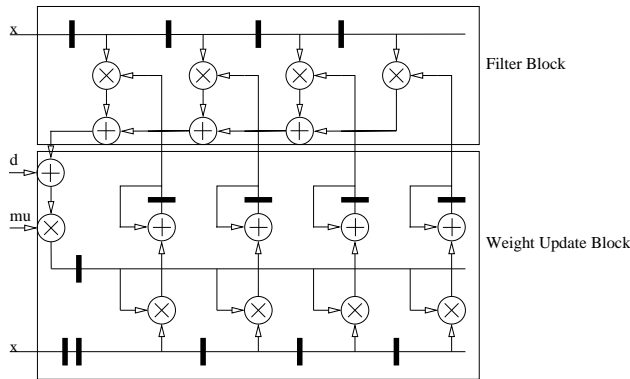


Figure 6: Direct form DLMS structure

filter block identical to the transposed FIR with fixed coefficients shown earlier. Combining the direct and trans-

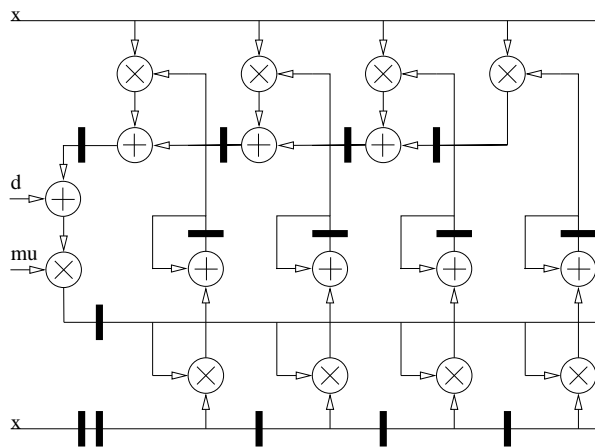


Figure 7: Transposed DLMS

posed form of the DLMS give a hybrid structure shown in Figure 8. In VLSI implementation, hybrid forms can be valuable since they are faster than direct forms and can easily be pipelined. The intention of this work was to see if similar remarks would apply for FPGA implementation.

4 Implementation

A variety of field programmable gate array architectures are available from various vendors. We evaluate our designs using the Xilinx Virtex series as this provides good low power, high complexity performance suitable for datapath operations. In Virtex designs, the basic building block of each configurable logic block (CLB), which contributes to providing the functional elements for constructing logic within the FPGA, is the logic cell (LC).

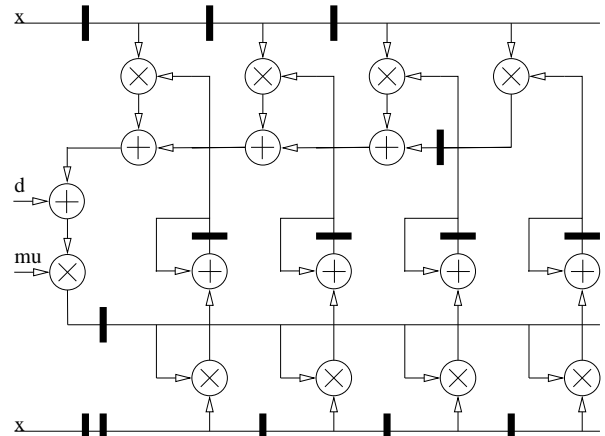


Figure 8: Hybrid DLMS

With the register rich architecture, pipelining can occur at this LC level. Where arithmetic functions are to be implemented, dedicated carry logic provides fast arithmetic carry capabilities for high speed arithmetic functions. The Virtex CLB supports two separate carry chains (one per slice), which allows the implementation of ripple carry adders (RCA) at minimal cost. To add two n bit numbers with a carry save adder we need n full adders followed by n ripple carry adders [4]. Its highly regular structure makes it ideal for implementation on an FPGA. Without the presence of dedicated carry logic, it would be too slow due to the fact the output would have to ripple through the full n RCA's. With the Virtex, losses in comparison with a much faster adder such as the carry look ahead adder (CLA) are minimal. This is also due to the highly irregular structure of the CLA which uses a generate and a propagate term to create the carry. It requires $\log_2 n$ logic levels as opposed to $2n$ logic levels in the case of a carry save adder. Implementing both these on a Virtex XCV1000E, gives a maximum combinational path delay of 14.395ns for a 12 bit carry save adder as opposed to 17.193ns with a 12 bit carry look ahead adder. The use of carry save arithmetic also lends itself to the improvement of multiplier structures. In addition, a dedicated AND gate within an LC improves the efficiency of multiplier implementation. Investigating methods of power saving led to use of a Booth multiplier [4], which halves the number of partial products in the multiplier making it faster and smaller.

5 Results

Our goal is to investigate the trade-off between these various possible architectures and compare them in terms of maximum speed, relative power and resource usage. Filters have been implemented using fixed point twos complement integer arithmetic. Input/output data and coeffi-

coefficients are twelve bits wide and filter lengths investigated are four, eight and sixteen taps. Input data was limited to integers in the range -5 to +5 ensuring no overflow. A step size of the power of two is used in these filters, 2^{-7} for these tests, as it reduces the scaling multiplication to a shifting operation. Verilog descriptions were synthesised to the target FPGA - a Virtex XCV1000E-8. Table 1 shows the achievable clock speeds for the de-

Table 1: Virtex Implementation (post layout) results for system clock rate (MHz)

	Direct form	Transposed form	Hybrid Form
4	22.98	48.20	39.22
8	18.13	47.33	40.31
16	11.15	47.63	39.10

sign. System performance can reach up to 48MHz for the transposed form. Maximum frequency remains relatively constant for the transposed form over 4, 8 and 16 taps, with a significant increase in speed over the direct form. The critical path is only one adder and one multiplier, independent of the number of taps. In ASIC design as the number of taps is increased in the transposed form, the capacitance of the input bus would limit its speed, and the advantages of the hybrid form, would become more evident. However as can be seen from the

Table 2: Area in terms of CLB's

	Direct	Transpose	Hybrid
4 taps	667	680	678
8 taps	1403	1417	1408
16 taps	2887	2894	2893

results in Table 1 this does not appear to be a problem with FPGA implementation for up to a 16 tap filter. In contrast the performance of the direct form implementation falls from almost 23MHz for a 4 tap filter to 11MHz for a 16 tap filter due to the extension in critical path. There is a 50% drop between 4 and 16 taps. The speed in the transposed form is achieved at a slight expense of area, which can be seen in Table 2. Table 3 indicates the estimated power for the designs considered. Their power was estimated using the Xilinx Virtex Power estimator Worksheet V.1.5. However these figures represent estimated datapath power only and do not take into account interconnect power which is usually far more significant. The hybrid form provides a compromise between the direct and hybrid forms, as can be seen from Tables 2 and 3.

6 Conclusion

In this paper alternative implementations of the delayed LMS adaptive algorithm are investigated. As expected,

Table 3: CLB power for 16 taps

	Direct	Transpose	Hybrid
CLB power	46mW	48mW	47mW

the direct form implementation is slowest, with its maximum frequency falling as the critical path increases. In contrast the maximum frequency of the transposed and hybrid forms are approximately independent of the number of taps due to the fact their critical paths are restricted to a fixed number of multipliers and adders. The potential fan in capacitance limited speed of the transposed form, is not a factor for filters of the size we have investigated. The hybrid form performs quite well in terms of power consumption and area, with its speed remaining relatively constant over the full range of taps. Its modular structure gives it a distinct advantage in terms of speed over the direct form structure. Additional pipelining would further improve results provided an appropriate pipelined architecture was chosen, so as not to give the inevitable huge increase in FPGA area which would be expected. However, there is no particular benefit in choosing hybrid form implementations in preference to transposed form for the cases investigated.

References

- [1] K. Azadet and C.J. Nicole. Low power equalizer architectures for high speed modems. *IEEE Communications Magazine*, pages 118–126, October 1998.
- [2] S.C. Douglas, QH Zhu, and K.F. Smith. A pipelined lms adaptive fir filter architecture without adaptation delay. *IEEE Transactions on Signal Processing*, 46(3):775–779, March 1998.
- [3] Long G, F. Ling, and J.G. Proakis. The lms algorithm with delayed coefficient adaptation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(9):1397–1405, September 1989.
- [4] J. Hennessey and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 1990.
- [5] H.R. Lee, C.W.Jen, and C.M. Liu. A new hardware efficient architecture for programmable fir filters. *IEEE Transactions Circuits and Systems II Analog and Digital Signal Processing*, 43:637–644, 1996.
- [6] S. Mitra and K. Kaiser. *Handbook for Digital Signal Processing*. Wiley, 1993.
- [7] N. Shanbhag and K. Parhi. *Pipelined Adaptive Digital Filters*. Kluwer, 1994.